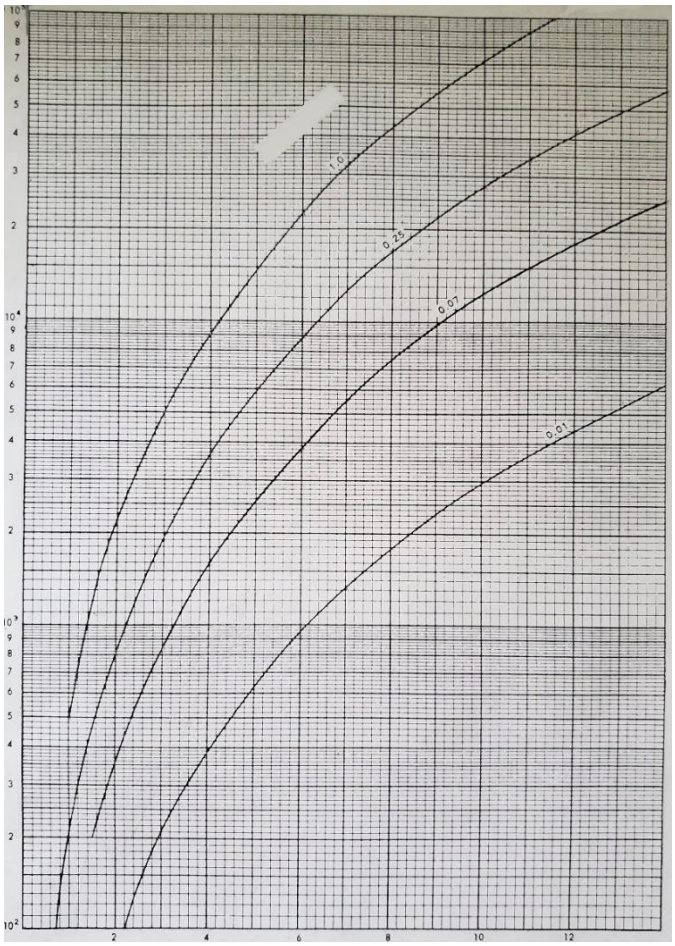
Case Study

Objective:

To computerize a plot with 4 curves to allow a user to solve for points not necessarily along those lines. The curves represent optimal flow rates for fluids of four different densities over ranges of pipe diameters. So the flow is presented as a function of pipe diameter and fluid density.



Method

1. Create a list of diameter values along the horizontal axis (you will be assigning the vertical axis values to these points in the next step). Use points from beginning of data given to the end with enough points between to be able to create an accurate plot. For this example we used 13 fairly evenly spaced points between 2.1 (the lower limit that we had data points for all of our curves) and 14 (the upper limit that we had data points for all of our curves). We named this list "DiaList".

2. Create another list for each of the different curves of the corresponding vertical value for each point in the list created above. In our case we created 4 lists and we named the D01, D07, D25 and D10 (corresponding to the curves of .01, .07, .25 and 1.0).

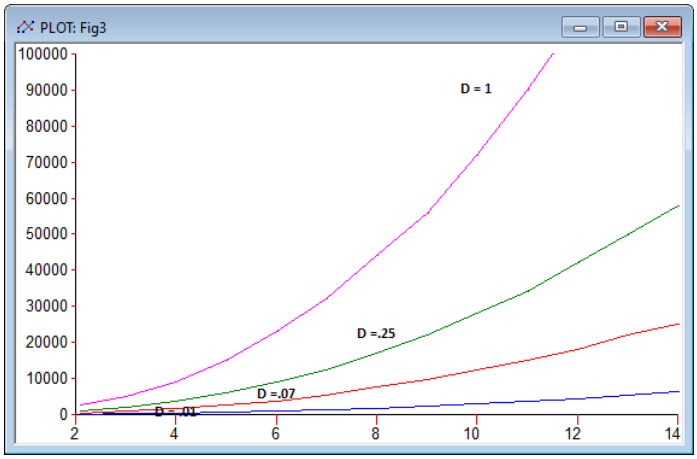
3. You can create a table using those lists. This will show the data points you collected along each of the designated lines of the original plot.

   As you can see, on our curve of D=.07 we have a value of 9,800 for our DiaList value of 9.

INTERACTIVE TABLE: Fig3

Element	DiaList	D01	D07	D25	D10
1	2.1	100	440	1000	2700
2	3	210	840	1900	5000
3	4	380	1600	3700	9000
4	5	630	2700	6000	15000
5	6	940	3800	8900	23000
6	7	1300	5500	12500	32000
7	8	1800	7600	17000	44000
8	9	2300	9800	22000	56000
9	10	3000	12250	28000	72000
10	11	3600	15000	34000	90000
11	12	4400	18000	42000	110000
12	13	5200	22000	50000	140000
13	14.0000001	6200	25000	58000	170000
14					
15					

4. Now create a plot using those lists. This allows you to see your curves to determine if you have enough plot points to keep your curves accurate.

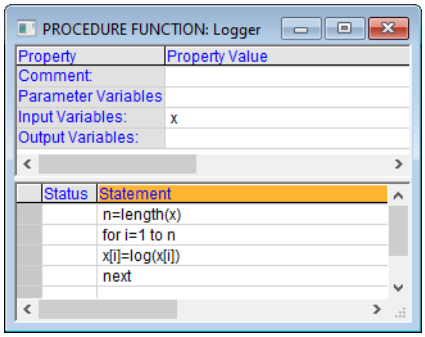


Notice this does not look like our original plot. This is because when determining the data points along the y axis, we did not use log values.

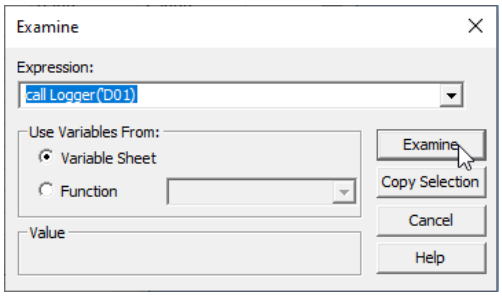
As you can see from this plot the data points along the y axis get large very fast. This is why we can get a better picture of what is going on if we convert these values to log scale.

5. We will now convert our lists of vertical data points to log.

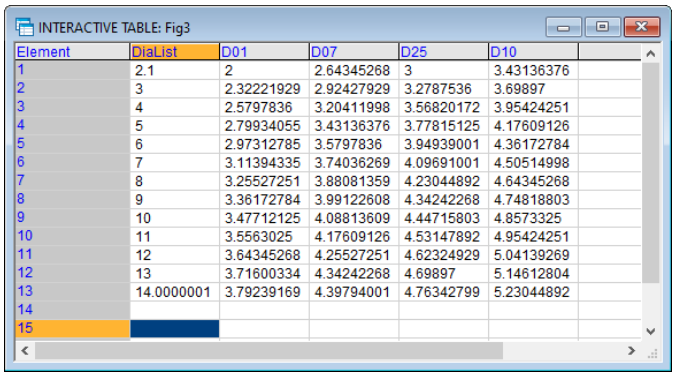
We do this by creating a small Procedure Function. We called this function “Logger”. It looks like this:



We run this function on lists D01, D07, D25 and D10 using the “Examine” command:

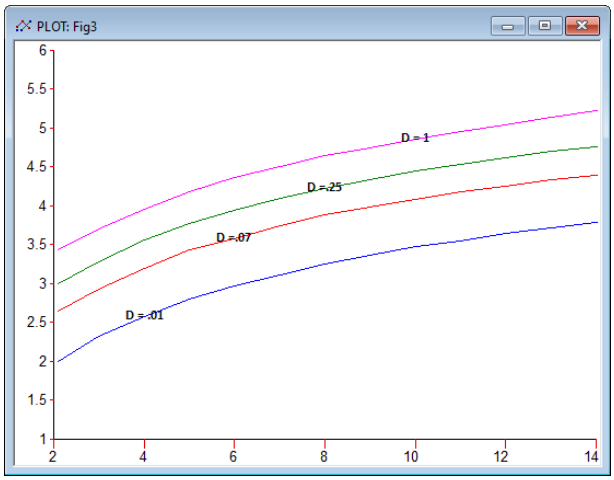


Now our data values looks like this. Note that now on our curve of D=.07 we have a value of 3.99 for our DiaList value of 9.



Element	DiaList	D01	D07	D25	D10
1	2.1	2	2.64345268	3	3.43136376
2	3	2.32221929	2.92427929	3.2787536	3.69897
3	4	2.5797836	3.20411998	3.56820172	3.95424251
4	5	2.79934055	3.43136376	3.77815125	4.17609126
5	6	2.97312785	3.5797836	3.94939001	4.36172784
6	7	3.11394335	3.74036269	4.09691001	4.50514998
7	8	3.25527251	3.88081359	4.23044892	4.64345268
8	9	3.36172784	3.99122608	4.34242268	4.74818803
9	10	3.47712125	4.08813609	4.44715803	4.8573325
10	11	3.5563025	4.17609126	4.53147892	4.95424251
11	12	3.64345268	4.25527251	4.62324929	5.04139269
12	13	3.71600334	4.34242268	4.69897	5.14612804
13	14.0000001	3.79239169	4.39794001	4.76342799	5.23044892
14					
15					

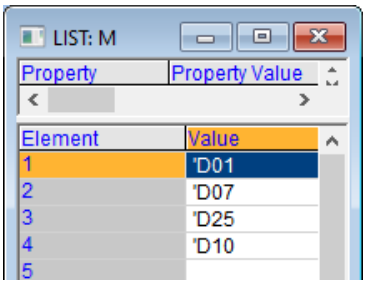
Now our plot looks like the original:



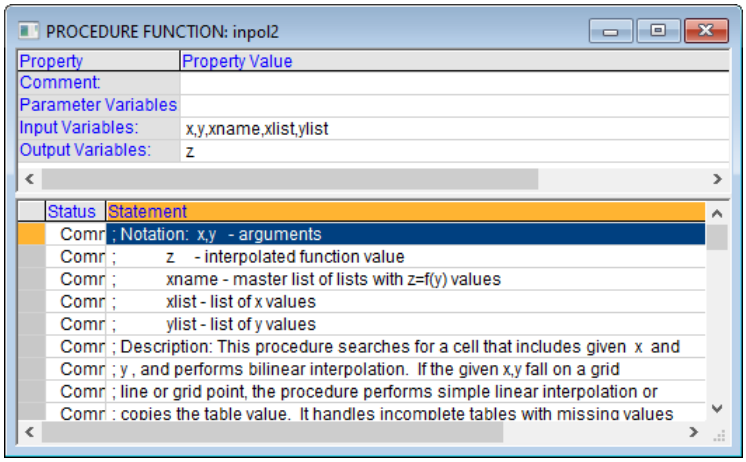
6. Now we will create one list containing the values for each of the 4 curves. For our example the curves had values of .01, .07, .25 and 1.0. We named this list “Xlist”.

   We will convert this lists to log values also since our other data points are log values.

7. Create one list that contains the names of the above lists. We named this list “M”. This is a matrix in TK Solver. This matrix will be called in the function we will import in the next step.



8. Import the bilinear interpolation tool (inpol2) from the TK Solver Library (found in the Utilities – Model Solving – Tools folder). This function will use your lists of data points and interpolate within those lists to solve for values between the known data points.



9. Call inpol2 from your rule sheet using the 5 input variables specified in the function. We are bringing in our Matrix and the other 2 lists associated with those data points.

   z = 10^(inpol2(log(x),y,'M,'Xlist,'DiaList))

   Since we converted the data points to log we also converted the “x” variable to log in this rule.

10. Import the bisecting tool (bisect) from the TK Solver Library (found in the Mathematics – Roots of Equations – Tools folder).

    At this point your model will solve if you know x and y. But what if you know z and x or z and y? Your rule on the rule sheet cannot be backsolved so it will not solve for x or y. This is where the bisect tool will help. This function will allow us to backsolve by using the bisection method to find the roots of an equation. The equation you will be finding roots for and the call to the bisect function are explained below.

11. We will create an equation that solves for the amount of error. We will start with the equation we know:

    z = 10^(inpol2(log(x),y,'M,'Xlist,'DiaList))

    If this is not true then we can add an amount “error” to make it true. We now have:

    z = 10^(inpol2(log(x),y,'M,'Xlist,'DiaList)) + error

    or:

error = z - 10^(inpol2(log(x),y,'M,'Xlist,'DiaList))

12. Now call the bisect function from the rule sheet specifying the bounds you will be solving in. In this case we are using the lower limit of 2.1 and the upper limit of 14.

    y=bisect('obj, 2.1,14)

13. Now that everything solves within the bounds of our data, we will add error messages to ensure our input stays within the bounds of our data:

    if y<2.1 then call errmsg("Pipe Diameter must be between 2.1 and 14")
    if y>14 then call errmsg("Pipe Diameter must be between 2.1 and 14")
    if x<.01 then call errmsg("Density must be between .01 and 1")
    if x>1 then call errmsg("Density must be between .01 and 1")